A Fine OTP Server

April 11, 2017

- Cryptography, RSA
- 79 points
- Connect to OTP generator server, and try to find one OTP. nc 66.172.27.77 35156

  So in the OTP server, they give us a RSA modulus, an encrypted message, and the function used to encrypt the message. The One Time Pads are encrypted with:

  ```
  def gen_otps():
  template_phrase = 'Welcome, dear customer, the secret passphrase for today is: '
      OTP_1 = template_phrase + gen_passphrase(18)
      OTP_2 = template_phrase + gen_passphrase(18)
      otp_1 = bytes_to_long(OTP_1)
      otp_2 = bytes_to_long(OTP_2)
      nbit, e = 2048, 3
  privkey = RSA.generate(nbit, e = e)
      pubkey  = privkey.publickey().exportKey()
      n = getattr(privkey.key, 'n')
      r = otp_2 - otp_1
  if r < 0:
          r = -r
  IMP = n - r**(e**2)
      if IMP > 0:
       c_1 = pow(otp_1, e, n)
       c_2 = pow(otp_2, e, n)
      return pubkey, OTP_1[-18:], OTP_2[-18:], c_1, c_2
  ```

The template phrase is 60 bytes long, and OTP_1 which includes the passphrase is 78 bytes, or 624 bits long. $OTP_1^3$is on the order of $624*3$ bits long. $624*3 = 1872 < 2048$ bits, so this doesn't wrap around the modulus! We just need to take the cubed root of one of the given messages, and provide that to the server to get the flag.

```
valar@valardev-Vostro-3460-mint ~ $ nc 66.172.27.77 35156
```

```
|-------------------------------------|
| Welcome to the S3cure OTP Generator |
|-------------------------------------|
| Guess the OTP and get the nice flag!|
| Options:
  [F]irst encrypted OTP
  [S]econd encrypted OTP
  [G]uess the OTP
  [P]ublic key
  [E]ncryption function
  [Q]uit
P
the public key is:
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAvdBapg5SXCJHVikgokU0 c0LA67ftF9ZhIrqSETuq3N
nQIBAw==
-----END PUBLIC KEY-----
S 1342484916452752140375644505087019657103834926373832886072831761324991239454706093232
Now if we just take the cubed root in python, with the following:
>>> import gmpy
>>> S = 13424849164527521403756445050870196571038349263738328860728317613249912394547706
>>> a = 23766750386011171096524965335062551726344030679891013787685943574919990535673610
>>> import binascii
>>> binascii.unhexlify(hex(a)[2:-1])
’Welcome, dear customer, the secret passphrase for today is: UEcAoQ9pGZ16DCWPPi’
```

If we enter UEcAoQ9pGZ16DCWPPi into our netcat session, we get our flag:

ASIS{<some random hex here>}